book is misleading. While the title of the original French version, *Aspects Mathématiques des Réseaux de Petri*, was vague but not wrong, the English translation, *The Mathematics of Petri Nets*, is annoyingly inappropriate. The intersection of mathematics and Petri nets is not restricted to computability and formal language theory; it also covers complexity, program semantics and verification, category theory, and other subject matters. Petri nets have profited from advances in these areas, and also contributed with indigenous techniques to them.

Javier Esparza
Institut für Informatik
Universität Hildesheim
Hildesheim, Germany

ABCL: An Object-Oriented Concurrent System. Edited by Akinori Yonezawa. MIT Press, Cambridge, MA, 1990, Price £35.95 (hardback), ISBN 0-262-24029-7.

In June 1988, I reviewed in this journal a previous book in the MIT Press Computer Systems Series, called *Object-Oriented Concurrent Programming*, edited by Akinori Yonezawa and Mario Tokoro. In that review, I observed that the meaning of the term "object-oriented" was not widely understood, with a great many different ideas about just what "objects" were. The previous volume reported on work on *concurrent* object-oriented systems from various research institutions around the world. This work was being actively pursued, with the hope that practical concurrent object-oriented programming languages could effectively exploit large-scale parallel computer hardware.

Since that time, there has been much clarification in the software engineering world about what is meant by an object-oriented system. In general, the view is that identifiable objects encapsulate data, and each object is an instance of a class, which represents the code shared by each of its instances. Individual methods or functions defined within a class are activated by sending a message to an instance of that class; this is roughly equivalent to a function call, except that the function body is determined dynamically, at the time the function is actually called. Furthermore, classes are related by inheritance, so that the behaviour of a class of objects can be refined in subclasses. In such systems, most objects are passive, with only a small amount of concurrent activity, which is programmed explicitly. There is now a considerable amount of commercial interest in object-oriented design and analysis, and in production-quality object-oriented programming languages. This approach can now be regarded as part of the mainstream of development approaches for applications on uniprocessors and small-scale parallel systems.

Conversely, work on concurrent object-oriented systems aimed at exploiting massively parallel computers has gone rather quiet. Such systems have a large number of objects which execute independently, in parallel. Between such objects,

a message-send is more akin to an inter-process communication, which may of course traverse a communication network connecting many processors. Many such systems do not use the notion of a class, but each object has its own private code; typically, some other mechanism, such as delegation, is used to share behaviour between objects. While there are a small number of commercial concurrent object-oriented languages, it is generally accepted that the problems associated with concurrent object-oriented systems are harder than was at first thought, and solutions are still being sought, so there is still active research work at various places around the world.

In the preface, Akinori Yonezawa observes that this new book is a sequel to the previous volume, but that it concentrates on work on just one concurrent object-oriented language, called ABCL/1. The work on this system is based primarily at the University of Tokyo, under the direction of Yonezawa. The book contains a great deal of detailed information about ABCL/1 and its associated computational model (ABCM/1).

The book is organised as a collection of independent chapters by different authors. Many of the chapters are revised versions of papers which have appeared elsewhere, notably in the proceedings of the annual ECOOP (European Conference on Object-Oriented Programming) and OOPSLA (Object-Oriented Programming: Systems, Languages and Applications) conferences. Various chapters describe theoretical aspects of the computational model, some details for the more interesting aspects of the implementation, in particular, debugging a distributed parallel application, and some example applications. An extensive appendix contains a user's guide for ABCL/1. A uniprocessor version of the ABCL/1 system is available free of charge; instructions on getting a distribution are included at the back of the book.

In conclusion, this is not a book for the reader merely slightly interested in the topic. Rather, this is a book recommended for the serious student of concurrent object-oriented systems, a reader who is prepared to put some significant effort in understanding the issues and problems. I would echo Yonezawa's sentiments where he says, in the preface, that he hopes that "... this publication would further stimulate the research and development of object-oriented concurrent computing".

Trevor P. HOPKINS
Computer Science Department
University of Manchester
Manchester, UK